Creating iPhone Apps That Communicate with Accessories

Session 507

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

At the convention center, a movie will be placed here

Creating iPhone Apps That Communicate with Accessories

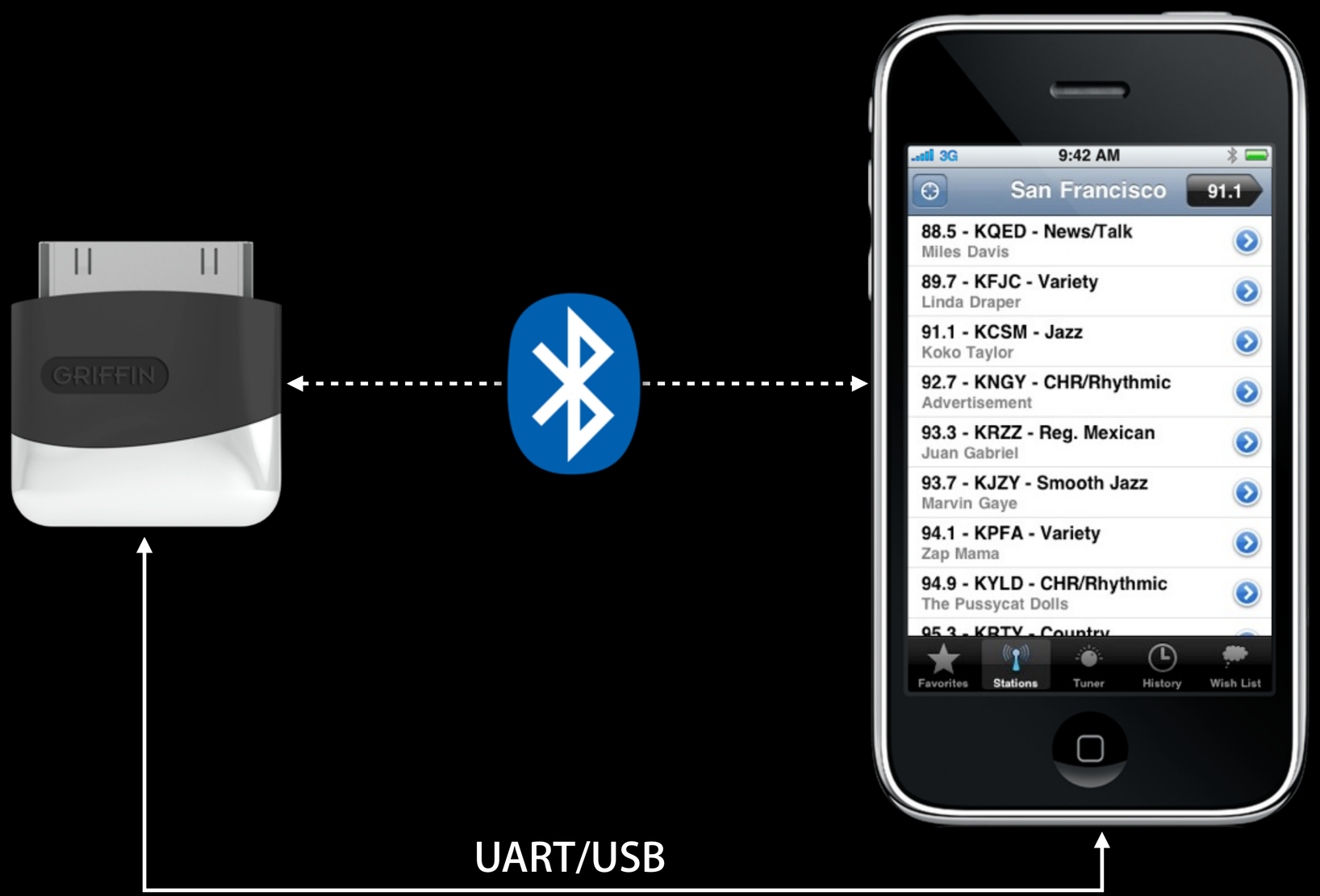
Paul Holden
iPhone Applications Engineer

Put Your Pencils Down…

- Demo source code available here
  - EA_Framework_demo

Who Are You?

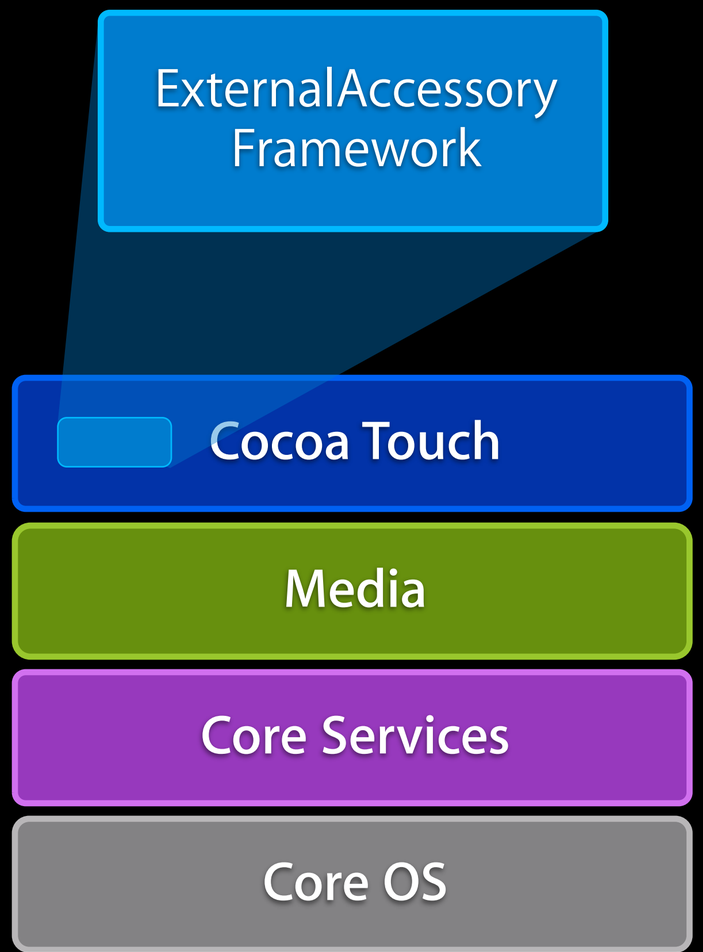
- Developers
  - MFi and WWi
  - iPhone app
  - Hardware


GRIFFIN
3G
9:42 AM
San Francisco
91.1
88.5 - KQED - News/Talk
Miles Davis
89.7 - KFJC - Variety
Linda Draper
91.1 - KCSM - Jazz
Koko Taylor
92.7 - KNGY - CHR/Rhythmic
Advertisement
93.3 - KRZZ - Reg. Mexican
Juan Gabriel
93.7 - KJZY - Smooth Jazz
Marvin Gaye
94.1 - KPFA - Variety
Zap Mama
94.9 - KYLD - CHR/Rhythmic
The Pussycat Dolls
Favorites
Stations
Tuner
History
Wish List
UART/USB

A Closer Look…

iPod Accessory Protocol

ExternalAccessory Framework



What You'll Learn

- The ExternalAccessory (EA) architecture
- Working with EA accessories in apps
  - Associating
  - Discovering
  - Communicating
- Protocol design guidelines

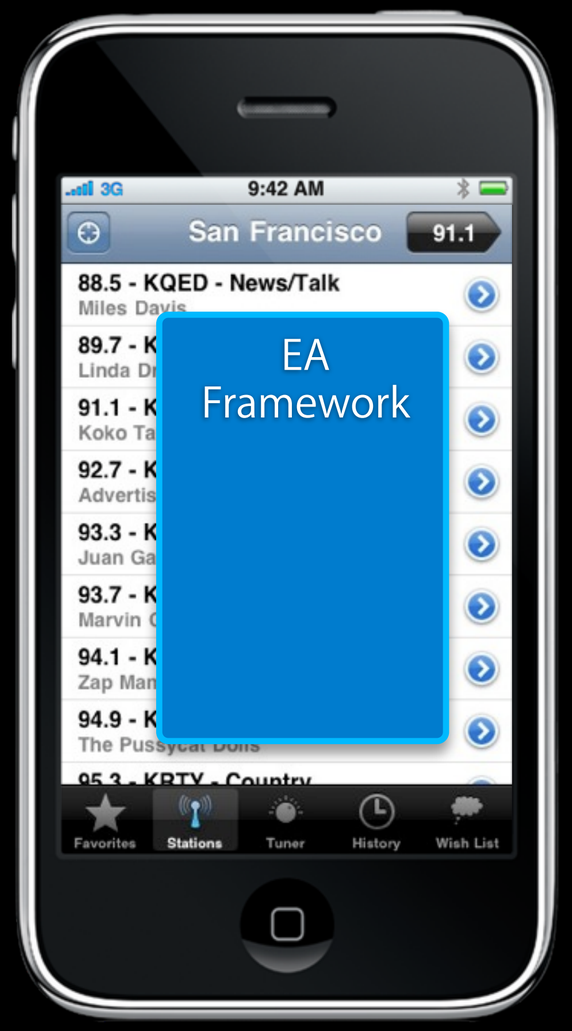
Developing Hardware

- Must participate in MFi Licensing Program
  - Technical documentation and support
  - Hardware components
  - iPod Accessory Protocol (iAP)

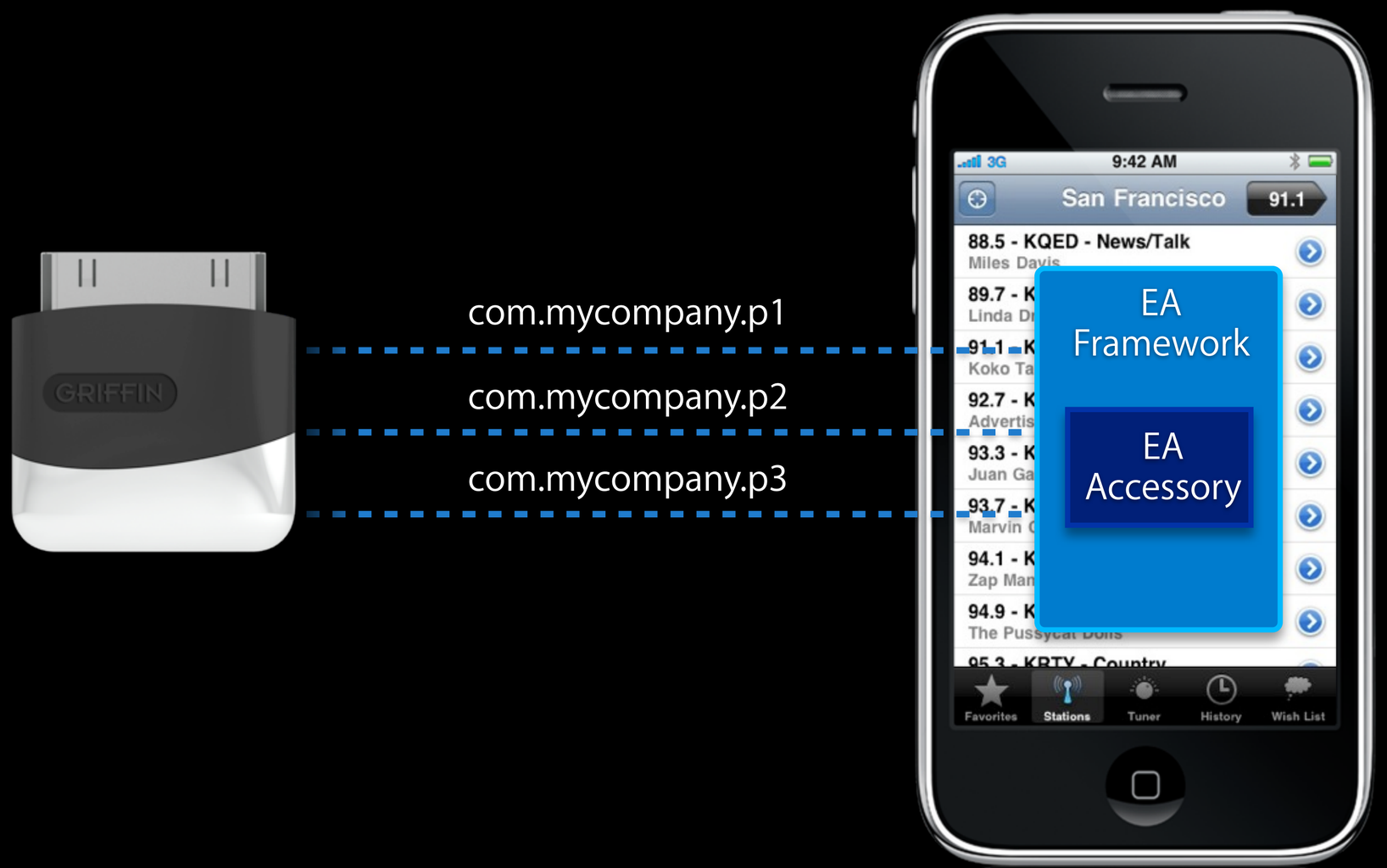
- For more information
  - http://developer.apple.com/ipod

Feature Description

The EA Architecture

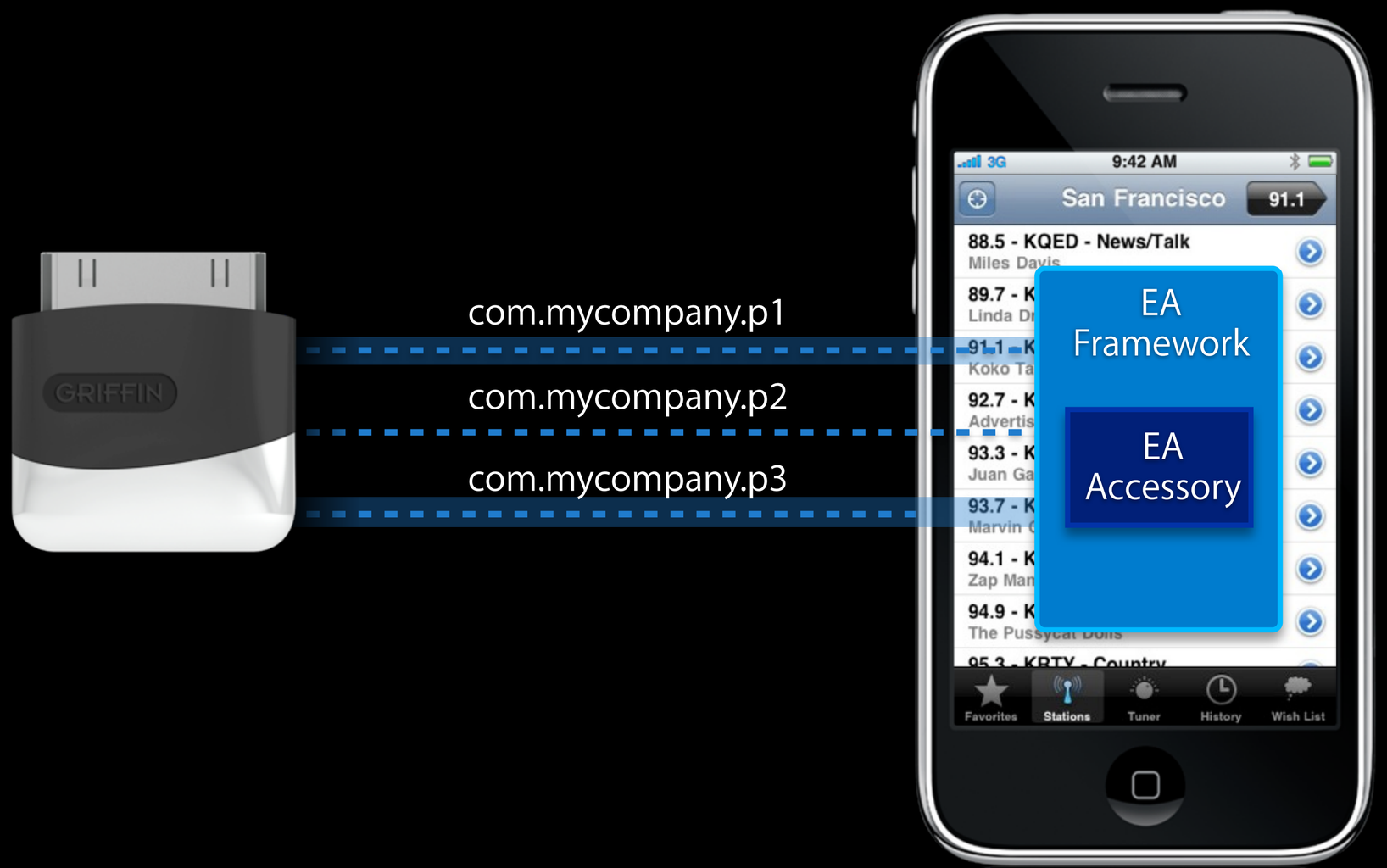


The EA Architecture—Connect



EA Protocol

The EA Architecture—Data Transfer



EA Protocol

EA Protocol with an associated EA session

What Is an EA Accessory

- Dock connector accessories (UART or USB)
- Bluetooth accessories

- Must be MFi / WWi
  - http://developer.apple.com/ipod

com.mycompany.p1

com.mycompany.p2

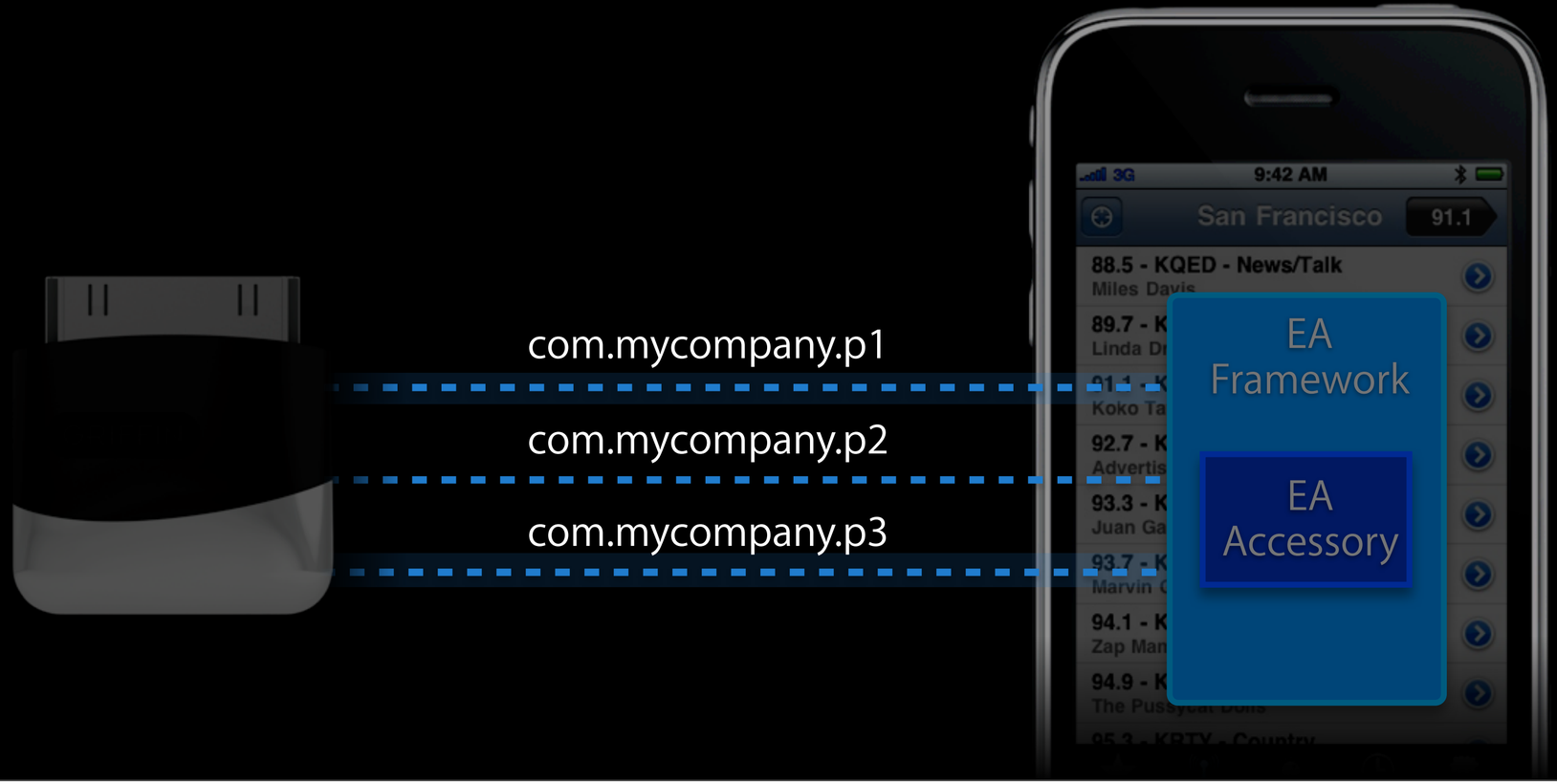
com.mycompany.p3

EA Protocol

- Definition… you tell me
- Logical communication channel
- Not related to `@protocol`
- Enumerated by:
  - Accessory
  - App

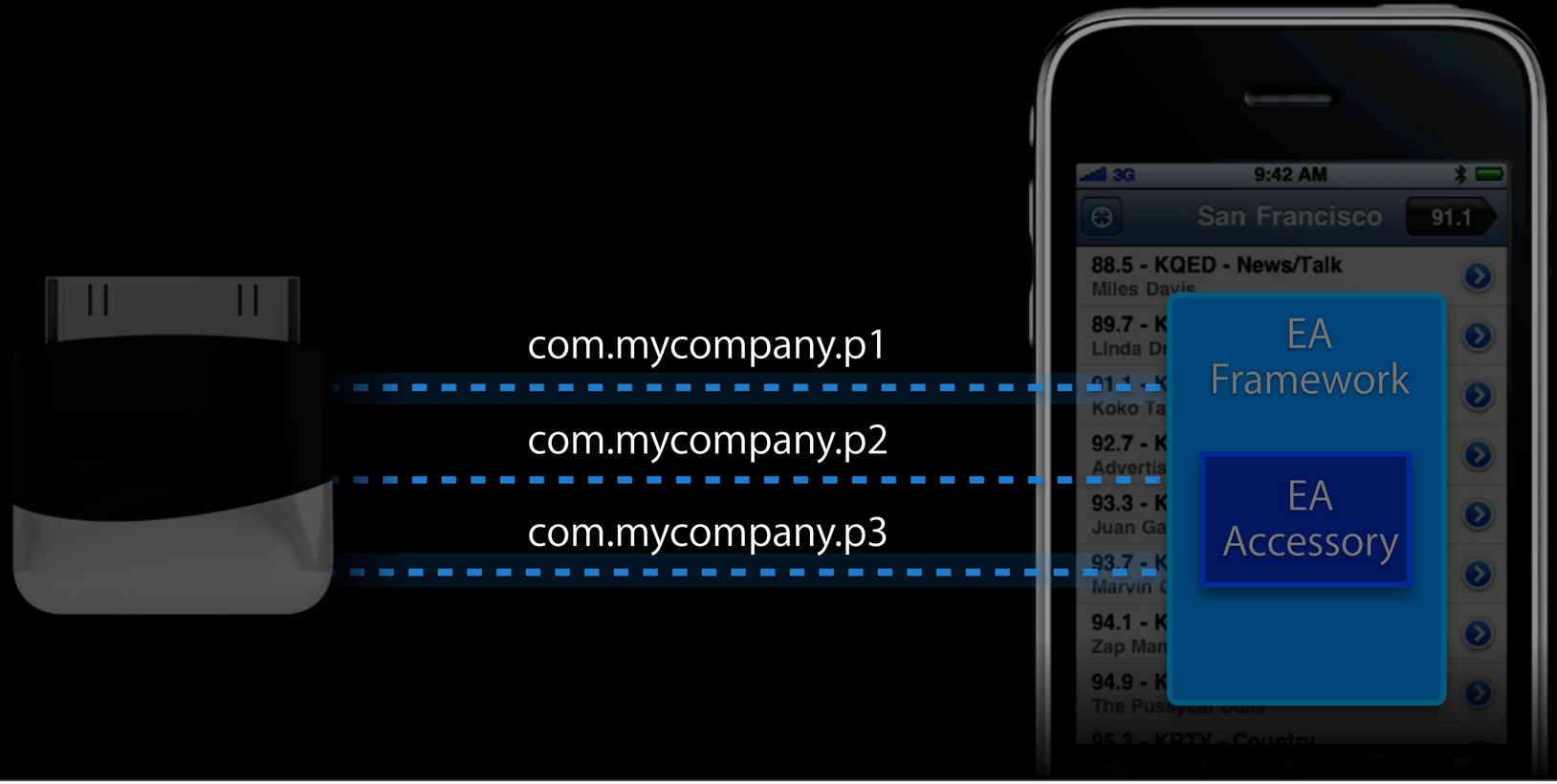
EA Protocol Rules

- Communication happens on an EA Protocol
- Name in reverse-DNS
  - `com.yourCompany.protocolName`



EA Protocol Visibility

- You decide
- You must enforce



EA Session

- Container for one EA Protocol object
  - Input/Output stream objects
- Only apps can create EA Sessions
- Apps create an EA Session to communicate over an EA Protocol

Associating Apps with EA Protocols

Add key to Info.plist

```
<plist version="1.0">
<dict>
    <key>CFBundleDevelopmentRegion</key>
...
    <key>UISupportedExternalAccessoryProtocols</key>
    <array>
        <string>com.apple.p1</string>
        <string>com.apple.p2</string>
    </array>
</dict>
</plist>
```

Associating Apps with EA Protocols

Info.plist… in Xcode

Key	Value
▼ Information Property List	(13 items)
Localization native development region	English
Bundle display name	${PRODUCT_NAME}
Executable file	${EXECUTABLE_NAME}
Icon file	icon.png
Bundle identifier	com.yourcompany.${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	6.0
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0
Application requires iPhone environment	☑
Main nib file base name	MainWindow
▼ Supported external accessory protocols ▲▼	(2 items)
Item 1	com.apple.p1
Item 2	com.apple.p2

Why Associate Your App? (Part 1)

Why Associate Your App? (Part 2)

External Accessory Framework

External Accessory Framework

- EAAccessoryManager
- EAAccessory
  - EAAccessoryDelegate protocol
- EASession

EAAccessoryManager API

```
NSString *const EAAccessoryDidConnectNotification
NSString *const EAAccessoryDidDisconnectNotification


+ (EAAccessoryManager *)sharedAccessoryManager;


- (void)registerForLocalNotifications;
- (void)unregisterForLocalNotifications;


@property (nonatomic, readonly) NSArray *connectedAccessories;
```

NOTE: Assume there can be more than one element in the NSArray!

EAAccessory API

```
@property(nonatomic, readonly) NSUInteger connectionID;
@property(nonatomic, readonly) NSArray *protocolStrings;
@property(nonatomic, assign) id<EAAccessoryDelegate> delegate;
```

EAAccessoryDelegate API

```
@protocol EAAccessoryDelegate <NSObject>
@optional
- (void)accessoryDidDisconnect:(EAAccessory *)accessory;
@end
```

EASession API

- One EA Session per EA Protocol per EA Accessory

```
- (id)initWithAccessory:(EAAccessory *)accessory forProtocol:
(NSString *)protocolString;


@property (nonatomic, readonly) NSInputStream *inputStream;
@property (nonatomic, readonly) NSOutputStream *outputStream;
```

NSInputSteam/NSOutputStream

- Subclasses of NSStream
- “Introduction to Stream Programming Guide for Cocoa”
  - Available on ADC website

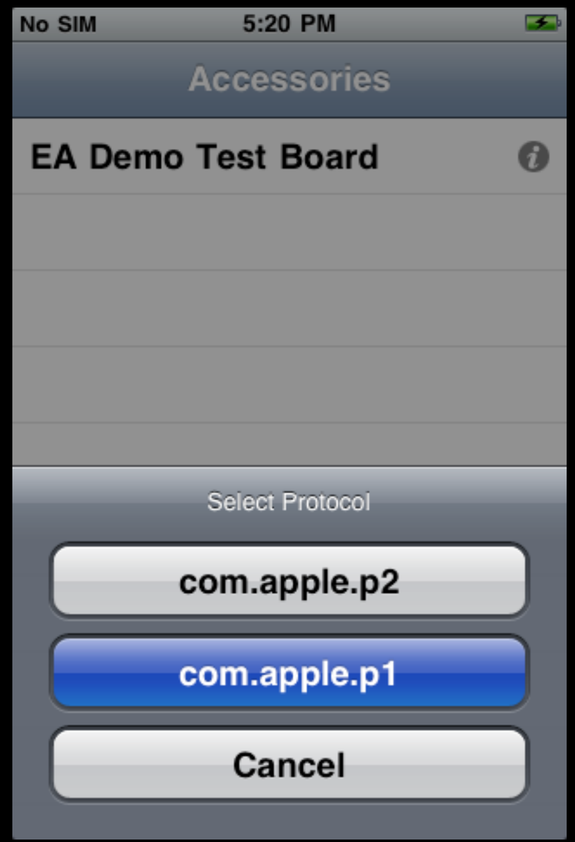
- Delegate handles stream events

```
- (void)stream:(NSStream *)theStream handleEvent:
(NSStreamEvent)streamEvent
```

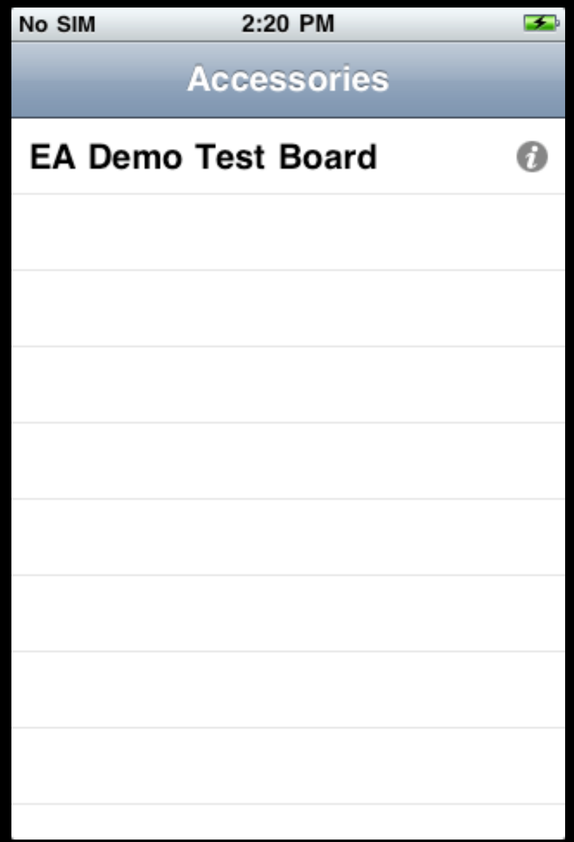
Demo

Simple LED app

EADemo



EADAccessoryDetailsViewController.h/m



EADAccessoryController.h/m

EADAccessoryController

EA Demo Test Board
com.apple.p1

```
- (void)writeData:(NSData *)data;
```

Let’s Look at the Code…

EA in the Simulator

Debugging Your Hardware

- For UART accessories, make the USB pins available

Designing an EA Protocol

- Data integrity
- Security
- Framing
- Future-proofing

Performance

- No latency guarantee
- Throughput
  - UART (19200 or 57600 bps)
  - USB
  - BT over RFCOMM

Goals

- Command and control
- Data exchange
- File transfer

Protocol Example

“WallBall”—Simple handball game for one

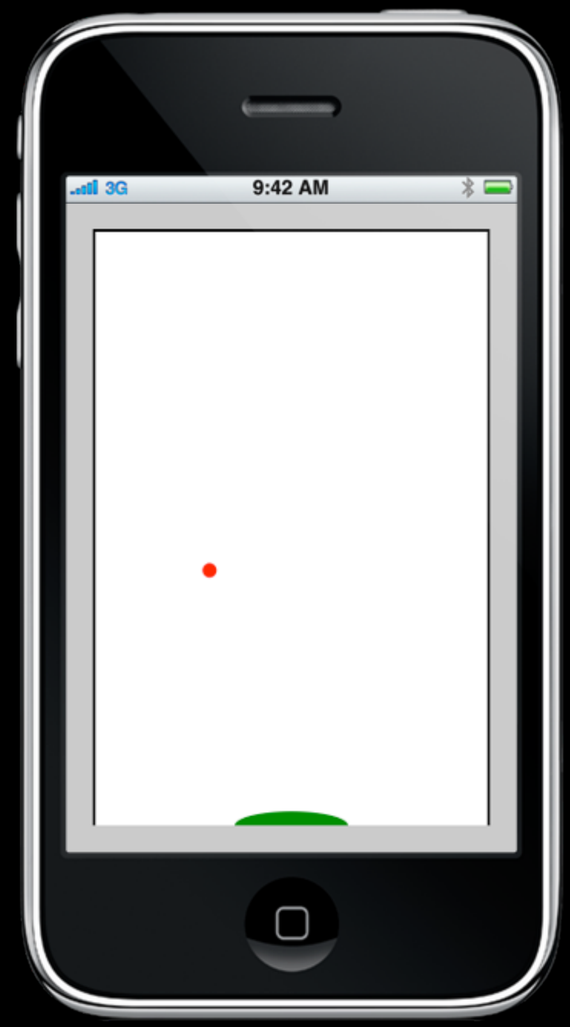
John Ananny
iPod and iPhone Accessory Engineer

WallBall Elements

- Physical accessory moves paddle
- Pushbutton changes paddle color
- Paddle hit flashes LED

- EA Accessory → App
  - Controller position updates
  - Pushbutton events

- App → EA Accessory
  - LED state



Let’s See It in Action…

WallBall EA Protocol

- Four-byte frames

bit:	7	6	5	4	3	2	1	0
byte 0	1	COMMAND[4:0]					PAYLOAD[15:14]	
byte 1	0	PAYLOAD[13:7]						
byte 2	0	PAYLOAD[6:0]						
byte 3	0	CRC[6:0]						

COMMAND	PAYLOAD	sent by
0	absolute controller position (0 == fully CCW)	accessory
1	bitmask of button states (only PAYLOAD[0] currently used)	accessory
2	bitmask of LED control states (only PAYLOAD[0] currently used)	app
3+	reserved	

WallBall EA Protocol

- com.apple.WallBall.paddle
- Achieved
  - Frame sync (MSb)
  - Data integrity (CRC)
  - Future expansion (reserved commands)
  - Low overhead
- No built-in security

Need Help?

More Information

Stephen Chick
iPhone Evangelism
chick@apple.com

Craig Keithley
iPod Technology Evangelist
keithley@apple.com

Documentation

iPhone OS Accessories
http://developer.apple.com/iphone/program/accessories/

Introduction to Stream Programming Guide for Cocoa
http://developer.apple.com/documentation/Cocoa/Conceptual/Streams/Streams.html

Labs

Creating iPhone Apps that Communicate with Accessories Lab	iPhone Lab B Wednesday 3:30PM

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09